

Steady-state parameter sensitivity in stochastic modeling via trajectory reweighting

Patrick B. Warren¹ and Rosalind J. Allen²

¹Unilever R&D Port Sunlight, Quarry Road East, Bebington, Wirral, CH63 3JW, UK.

²SUPA, School of Physics and Astronomy, The University of Edinburgh,
The Kings Buildings, Mayfield Road, Edinburgh, EH9 3JZ, UK.

(Dated: February 8, 2012)

Parameter sensitivity analysis is a powerful tool in the building and analysis of biochemical network models. For stochastic simulations, parameter sensitivity analysis can be computationally expensive, requiring multiple simulations for perturbed values of the parameters. Here, we use trajectory reweighting to derive a method for computing sensitivity coefficients in stochastic simulations without explicitly perturbing the parameter values, avoiding the need for repeated simulations. The method allows the simultaneous computation of multiple sensitivity coefficients. Our approach recovers results originally obtained by application of the Girsanov measure transform in the general theory of stochastic processes [A. Plyasunov and A. P. Arkin, J. Comp. Phys. **221**, 724 (2007)]. We build on these results to show how the method can be used to compute steady-state sensitivity coefficients from a single simulation run, and we present various efficiency improvements. For models of biochemical signaling networks the method has a particularly simple implementation. We demonstrate its application to a signaling network showing stochastic focussing and to a bistable genetic switch, and present exact results for models with linear propensity functions.

PACS numbers: 87.18.Tt, 87.10.Mn, 02.50.Ga

I. INTRODUCTION

Parameter sensitivity analysis is one of the most important tools available for modelling biochemical networks. Such analysis is particularly crucial in systems biology, where models may have hundreds of parameters whose values are uncertain. Sensitivity analysis allows one to rank parameters in order of their influence on network behaviour, and hence to target experimental measurements towards biologically relevant parameters and to identify possible drug targets. For deterministic models, the adjunct ODE method provides an efficient way to compute the local sensitivity of a model to small changes in parameters. For stochastic models, however, parameter sensitivity analysis can be computationally intensive, requiring repeated simulations for perturbed values of the parameters. Here, we demonstrate a method, based on trajectory reweighting, for computing local parameter sensitivity coefficients in stochastic kinetic Monte-Carlo simulations without the need for repeated simulations.

Sensitivity analysis of biochemical network models may take a number of forms. One may wish to determine how a model's behaviour changes as a parameter is varied systematically within some range (a parameter sweep), its dependence on the initial conditions of a simulation, or its sensitivity to changes in the structure of the model itself (alternate mode-of-action hypotheses). In this paper, we focus on the computation of local parameter sensitivity coefficients. These coefficients describe how a particular output f of the model varies when the α -th parameter of the model, k_α , is varied by an infinitesimal amount, $k_\alpha \rightarrow k_\alpha + h$:

$$\frac{\partial f}{\partial k_\alpha} = \lim_{h \rightarrow 0} \frac{f' - f}{h}. \quad (1)$$

where f' is the output of the model computed in a system with k_α changed to $k_\alpha + h$. For deterministic models, where the dynamics of the variables $x_i(t)$ can be described by a set of deterministic ordinary differential equations (ODEs) $\partial x_i / \partial t = g(x_i, k_\alpha)$, differentiation of the ODEs with respect to k_α shows that the sensitivity coefficients $C_{i,\alpha} \equiv \partial x_i / \partial k_\alpha$ obey an *adjunct* set of ODEs,

$$\frac{\partial C_{i,\alpha}}{\partial t} = \sum_j \frac{\partial g}{\partial x_j} C_{j,\alpha} + \frac{\partial g}{\partial k_\alpha}. \quad (2)$$

These adjunct ODEs can be integrated alongside the original ODEs to compute the sensitivity coefficients “on the fly” in a deterministic simulation of a biochemical network.

Stochastic models of biochemical networks are (generally) continuous-time Markov processes [1] which are solved numerically by kinetic Monte-Carlo simulation, using standard methods such as the Gillespie [2] or Gibson-Bruck [3] algorithms. Replicate simulations will produce different trajectories; we wish to compute how the *average* value $\langle f \rangle$ of some function $f(t)$ of the model changes with the parameter k_α :

$$\frac{\partial \langle f \rangle}{\partial k_\alpha} = \lim_{h \rightarrow 0} \frac{\langle f \rangle' - \langle f \rangle}{h}. \quad (3)$$

where the averages are taken across replicate simulation runs. If one is interested in steady-state (*i.e.* time-independent) parameter sensitivities, the averages in Eq. (3) may instead be time averages taken over a single simulation run. Naïve evaluation of parameter sensitivities via Eq. (3) is very inefficient, since one is likely to be looking for a small difference between two fluctuating quantities. There are several existing approaches that get around this problem: spectral methods [4], a

method based on the Girsanov measure transform [5], and methods which re-use the random number streams [6]. In this paper, we develop a method based on trajectory reweighting, which is simple to implement in existing kinetic Monte Carlo codes and provides a way to compute steady-state parameter sensitivity coefficients “on-the-fly” in stochastic simulations of biochemical networks. The method provides an accessible alternative to the Girsanov measure transform pioneered by Plyasunov and Arkin [5]. Indeed several of our equations in Section II are equivalent to those Ref. 5. However, we go beyond previous work by showing in practical terms how the method can be implemented in standard stochastic simulation algorithms, extending the method to the computation of parameter sensitivities in the steady state, and showing how time-step preaveraging can be used to improve the efficiency of the calculations.

II. TRAJECTORY REWEIGHTING

The basic idea behind trajectory reweighting is as follows. In a kinetic Monte-Carlo simulation, for a given set of parameters any given trajectory has a statistical weight which measures the probability that it will be generated by the algorithm [7–9]; this weight can be expressed as an analytical function of the states of the system along the trajectory and of the parameter set. This analytical function also allows us to compute the statistical weight for this *same* trajectory, in a system with a *different* set of parameters: *i. e.* its weight in the ensemble of trajectories with perturbed parameters. This allows us in principle to compute the average $\langle f \rangle'$ in Eq. (3) for the perturbed parameter set, using only a set of trajectories generated with the unperturbed parameter set. For most applications this is inefficient, because the weight of a trajectory in the perturbed ensemble is typically very low, resulting in poor sampling. However, it turns out that trajectory reweighting does provide an effective way to compute local parameter sensitivity coefficients.

Trajectory reweighting for kinetic Monte-Carlo simulations

More specifically, let us consider a typical implementation of the Gillespie algorithm [2] (similar arguments apply to more recent algorithms, such as Gibson-Bruck [3]). Here, the state of the system is characterised by a set of discrete quantities N_i , typically representing the number of molecules of chemical species i . Transitions between states are governed by propensity functions $a_\mu(N_i, k_\alpha)$ where μ labels the possible reaction channels and the quantities k_α are parameters in the problem, typically reaction rates (k_α represents the α -th such parameter). A kinetic Monte-Carlo trajectory is generated by stepping through the space of states N_i in the following way. We first compute the propensity functions $a_\mu(N_i)$ for all

the possible transitions out of the current state. We then choose a time step (*i. e.* waiting time) δt from an exponential distribution $p(\delta t) = \tau^{-1}e^{-\delta t/\tau}$, where the state-dependent mean timestep (the mean waiting time before exiting the current state) is $\langle \delta t \rangle = \tau \equiv (\sum_\mu a_\mu)^{-1}$. We choose a reaction channel with probability $p_\mu = a_\mu / \sum_\mu a_\mu$. We advance time by δt and update the values of N_i according to the chosen reaction channel. We are now in a new state, and the above steps are repeated.

Now let us consider the statistical weight of a given trajectory generated by this algorithm [7–9]. In each step, the probability of choosing the value of δt that we actually chose is proportional to $\tau^{-1}e^{-\delta t/\tau}$, and the probability of choosing the reaction channel that we actually chose is equal to $a_\mu / \sum_\mu a_\mu$. We can therefore associate a weight P with the whole trajectory, which is proportional to the probability of generating the sequence of steps which we actually generated:

$$\begin{aligned} P &= \prod_{\text{steps}} (a_\mu / \sum_\mu a_\mu) \times (\tau^{-1}e^{-\delta t/\tau}) \\ &= \prod_{\text{steps}} a_\mu e^{-(\sum_\mu a_\mu)\delta t}. \end{aligned} \quad (4)$$

The second line follows by eliminating $(1/\sum_\mu a_\mu) \times \tau^{-1} \equiv 1$ (note that because Eq. (4) is not normalized, P is a *weight* rather than a true probability).

In a typical kinetic Monte-Carlo simulation, we generate multiple independent trajectories of length t , for a given parameter set. The probability of generating any given trajectory in this sample will be proportional to its weight P , defined in Eq. (4). We then compute the average $\langle f(t) \rangle$ of some function $f(t)$ of the state of the system by summing over the values of f , at time t , for these trajectories.

Having generated this set of trajectories, let us now suppose we wish to re-use them to compute the average $\langle f(t) \rangle'$ which we would have obtained had we repeated our simulations for some *other* parameter set. It turns out that we can compute this average by summing over the same set of trajectories, multiplied by the ratio of their statistical weights for the perturbed and unperturbed parameter sets. To see this, we first recall that an average, *e. g.* $\langle f(t) \rangle$, can be written as a sum over all *possible* trajectories j of length t , multiplied by their statistical weights P_j : $\langle f(t) \rangle = (\sum_j P_j f_j(t)) / (\sum_j P_j)$. Writing the perturbed average $\langle f(t) \rangle'$ in this way, we obtain

$$\langle f(t) \rangle' = \frac{\sum_j P'_j f_j(t)}{\sum_j P'_j} = \frac{\sum_j (P'_j/P_j) P_j f_j(t)}{\sum_j (P'_j/P_j) P_j} = \frac{\langle f(t) P'/P \rangle}{\langle P'/P \rangle} \quad (5)$$

where P and P' are the trajectory weights (calculated using Eq. (4)) for the original and perturbed models respectively. In another context, Eq.(5) has been used to reweight trajectory statistics in order to sample rare events in biochemical networks [9]; it also forms the basis of umbrella sampling methods for particle-based Monte Carlo simulations [10].

Whilst in principle Eq.(5) provides a completely general way to transform between trajectory ensembles with

different parameter sets, in practice it is useless for any significant deviation of the parameter set from the original values, for two reasons. First, the statistical errors in the computation of $\langle f \rangle'$ grow catastrophically with the size of the perturbation, because the original trajectories become increasingly unrepresentative of the perturbed model. Second, the computational cost of determining the trajectory weights for the perturbed and unperturbed parameter sets via Eq. (4) is only marginally less than the cost of computing $\langle f \rangle'$ directly by generating a new set of trajectories for the perturbed parameter set.

Computation of parameter sensitivity coefficients

It turns out, however, that Eq. (5) is useful for the computation of parameter sensitivity coefficients, where the deviation between the original and perturbed parameter sets is infinitesimal. Let us suppose that the perturbed problem corresponds to a small change in a single parameter, such as $k'_\alpha = k_\alpha + h$; the corresponding sensitivity coefficient is defined by Eq. (3). As we show in Supplementary Material Section 1, differentiating Eq. (5) with respect to k'_α leads to the following expression for the sensitivity coefficient:

$$\frac{\partial \langle f \rangle}{\partial k_\alpha} = \langle f W_{k_\alpha} \rangle - \langle f \rangle \langle W_{k_\alpha} \rangle. \quad (6)$$

where

$$W_{k_\alpha} = \frac{1}{P} \frac{\partial P}{\partial k_\alpha} = \frac{\partial \ln P}{\partial k_\alpha} \quad (7)$$

Supplementary Material Section 1 also shows how to generalize this approach to higher-order derivatives. Combining Eq. (4) with Eq. (7) shows that the “weight function” W_{k_α} can be expressed as a sum over all steps in the trajectory:

$$W_{k_\alpha} = \sum_{\text{steps}} \delta W_{k_\alpha} \quad (8)$$

where

$$\delta W_{k_\alpha} = \frac{\partial \ln a_\mu}{\partial k_\alpha} - \frac{\partial (\sum a_\mu)}{\partial k_\alpha} \delta t. \quad (9)$$

Eqs. (6)–(9) are the key results of this paper, since they point to a practical way to compute parameter sensitivity coefficients in kinetic Monte-Carlo simulations. To evaluate the (time-dependent) parameter sensitivity $\partial \langle f(t) \rangle / \partial k_\alpha$, one tracks a weight function $W_{k_\alpha}(t)$, which evolves according to Eqs. (8) and (9). One also tracks the function $f(t)$ of interest. The covariance between W_{k_α} and f , at the time t of interest, computed over multiple simulations, then gives the sensitivity of $\langle f(t) \rangle$ to the parameter in question (as in Eq. (6)). Tracking W_{k_α} should be a straightforward addition to standard kinetic Monte-Carlo schemes. Moreover we note that f could

be any function of the variables of the system—for example, if one were interested in the parameter sensitivity of the noise in particle number N_i , one could choose $f(N_i) = N_i^2$. More complex functions of the particle numbers, involving multiple chemical species, could also be used (see examples below).

This prescription for computing parameter sensitivities presents, however, some difficulties in terms of statistical sampling. The two terms in Eq. (9) are statistically independent quantities with the same expectation value, $\partial \ln(\sum a_\mu) / \partial k_\alpha$. Hence they cancel on average but the variances add. Thus we expect that W_{k_α} is a stochastic process with a zero mean, $\langle W_{k_\alpha} \rangle = 0$, and a variance that should grow approximately linearly with time—as shown for a simple example case in Supplementary Material Section 2—in effect W_{k_α} behaves as a random walk (*i. e.* a Wiener process). In terms of controlling the sampling error, this means that the number of trajectories over which the covariance is evaluated should increase in proportion to the trajectory length, since the standard error in the mean is expected to go as the square root of the variance divided by the number of trajectories [11]. In Section III, we discuss a way to avoid this problem, when computing steady-state parameter sensitivities.

Simplifications and practical implementation

Without loss of generality we can presume that the parameter k_α will appear in only one of the propensity functions, which we call a_α [12]. With this presumption, Eq. (9) becomes

$$\delta W_{k_\alpha} = \frac{\partial \ln a_\alpha}{\partial k_\alpha} (\delta_{\mu\alpha} - a_\alpha \delta t). \quad (10)$$

Eq. (10) makes a direct link with the Girsanov measure transform method introduced by Plyasunov and Arkin, being essentially the same as Eq. (31b) in Ref. 5.

A further simplification occurs if k_α is the rate coefficient of the α -th reaction, so that a_α is linearly proportional to k_α . One then has $\partial \ln a_\alpha / \partial k_\alpha = 1/k_\alpha$ and Eq. (8) becomes

$$W_{k_\alpha} = \frac{1}{k_\alpha} \left[Q_\alpha - \sum_{\text{steps}} a_\alpha \delta t \right] \quad (11)$$

where Q_α counts the number of times that the α -th reaction is visited. This is essentially the same as Eq. (9b) of Plyasunov and Arkin’s work, Ref. 5.

Eq. (11) suggests a very simple way to implement parameter sensitivity computations in existing kinetic Monte-Carlo codes. One simply modifies the chemical reaction scheme such that each reaction whose rate constant is of interest generates a “ghost” particle in addition to its other reaction products (this is similar to the clock trick in Ref. 13). There should be a different flavour of ghost particle for each reaction of interest, and ghost particles should not participate in any other reactions.

$Q_\alpha(t)$ is then simply given by the number of ghost particles associated with the α -th reaction which are present at time t . In Section IV, we use this approach to compute sensitivity coefficients using the *unmodified* COPASI [14] simulation package. In Supplementary Material Section 2 we also exploit this trick to obtain some exact results for linear propensity functions.

III. STEADY STATE

So far, we have discussed the computation of time-dependent parameter sensitivity coefficients $\partial\langle f(t) \rangle / \partial k_\alpha$, by evaluating the covariance of the weight function $W_{k_\alpha}(t)$ with the function $f(t)$ over multiple simulation runs. Often, however, one is interested in the parameter sensitivity of the *steady-state* properties of the system $\partial\langle f \rangle_{ss} / \partial k_\alpha$; this is a time-independent quantity. We now discuss the computation of steady-state parameter sensitivities using trajectory reweighting. We show that in this case, first, the problem of poor sampling of $W_{k_\alpha}(t)$ for long times can be circumvented, second, one can obtain sensitivity coefficients from a single simulation run, and third, one can improve efficiency by a procedure which we call time-step pre-averaging.

The ensemble-averaged correlation function method

To compute steady-state parameter sensitivities, one might imagine that we could simply apply the method discussed in Section II, taking the limit of long times, when the system should have relaxed to its steady state. However, this does not work, because the variance between trajectories of the weight function W_{k_α} increases in time, making it impossible to obtain good statistics at long times. To circumvent this problem, we note that the right hand side of Eq. (6) is unaltered if W_{k_α} is offset by a constant. Thus we may write the parameter sensitivity in the form of a two-point time-correlation function:

$$\frac{\partial\langle f(t) \rangle}{\partial k_\alpha} = C(t, t_0) = \langle f(N_i, t) \Delta W_{k_\alpha}(t, t_0) \rangle - \langle f(N_i, t) \rangle \langle \Delta W_{k_\alpha}(t, t_0) \rangle \quad (12)$$

where

$$\Delta W_{k_\alpha}(t, t_0) = W_{k_\alpha}(t) - W_{k_\alpha}(t_0), \quad (13)$$

and t_0 is some arbitrary reference time such that $t - t_0 = \Delta t > 0$. This relation has the advantage that we may choose Δt sufficiently small to make the variance of $\Delta W_{k_\alpha}(t, t_0)$ manageable. Importantly, in steady-state conditions, we expect that the correlation function depends only on the time difference and not separately on the two times, so that $C(t, t_0) = C(\Delta t)$, with $C(0) = 0$ and $C(\Delta t) \rightarrow \partial\langle f \rangle_{ss} / \partial k_\alpha$ as $\Delta t \rightarrow \infty$. Thus to calculate the sensitivity coefficient under steady state conditions, all we need to do is compute the steady-state

correlation function defined in Eq. (12), choosing a suitable ‘reference’ time t_0 when the system is already in the steady-state, then take the asymptotic (large Δt) value of this correlation function. We expect the correlation function to approach its asymptotic value on a timescale governed by the (likely short) relaxation time spectrum in the steady state, so that for most problems large values of Δt should not be required. Noting that in this method, as in Section II, the averages in Eq. (12) are computed over multiple independent simulation runs, we term this approach the *ensemble-averaged correlation function method*.

From a practical point of view, this method involves the following set of steps or ‘recipe’:

1. Choose two time points t_1 and t_2 such that the system has already reached its steady state at time t_1 and $t_2 = t_1 + \Delta t$ where Δt is greater than the typical relaxation time of the quantity f of interest (typically this is the same as the longest relaxation time in the system as a whole).
2. Compute W_{k_α} at times t_1 and t_2 and f at time t_2 .
3. Calculate the difference $\Delta W_{k_\alpha} = W_{k_\alpha}(t_2) - W_{k_\alpha}(t_1)$. Compute also the product $f(t_2) \Delta W_{k_\alpha}$.
4. Repeat steps 1-3 for many independent simulation runs and compute the averages $\langle f(t_2) \rangle$, $\langle \Delta W_{k_\alpha}(t_2 - t_1) \rangle$ and $\langle f(t_2) \Delta W_{k_\alpha}(t_2 - t_1) \rangle$ over the replicate simulations.
5. Calculate the correlation function $C(\Delta t) = \langle f \Delta W_{k_\alpha} \rangle - \langle f \rangle \langle \Delta W_{k_\alpha} \rangle$. As long as Δt is large enough this provides a measurement of $\partial\langle f \rangle_{ss} / \partial k_\alpha$.

The time-averaged correlation function method

It turns out, however, that for steady-state parameter sensitivities, we do not need to average over multiple simulation runs—we can instead compute time-averages over a single simulation run. This amounts to replacing the steady-state ensemble averaged sensitivity $\partial\langle f \rangle_{ss} / \partial k_\alpha$ by the *time averaged* version $\partial\bar{f}_{ss} / \partial k_\alpha$, where

$$\bar{f} = \frac{\sum_{\text{steps}} f \delta t}{\sum_{\text{steps}} \delta t} \quad (14)$$

(recalling that in kinetic Monte Carlo, the timestep δt varies between steps). In principle, $\partial\bar{f}_{ss} / \partial k_\alpha$ could be obtained by computing the time-averaged version of Eq. (12):

$$C_{\text{time av}}(\Delta t) = \overline{f(t) \Delta W_{k_\alpha}(t, t_0)} - \overline{f(t)} \overline{\Delta W_{k_\alpha}(t, t_0)} \quad (15)$$

and taking the limit of large $\Delta t = t - t_0$. Eq. (15) requires one to keep track of W_{k_α} a precise time Δt in the past; since the time step is not constant in kinetic Monte Carlo, this is rather inconvenient to implement.

Fortunately, however, tracking the weight function at a precise time in the past turns out to be unnecessary. As Δt becomes large, the stochastic differences between individual time steps cancel out and it becomes equivalent simply to compute the average

$$C_{\text{time av}}(n) = \overline{f(t) \Delta W_{k_\alpha}(n)} - \overline{f(t)} \overline{\Delta W_{k_\alpha}(n)} \quad (16)$$

where

$$\Delta W_{k_\alpha}(n) = W_{k_\alpha}(t) - W_{k_\alpha}(n \text{ steps ago}). \quad (17)$$

and to use the fact that $C_{\text{time av}}(n) \rightarrow \partial \bar{f}_{\text{ss}} / \partial k_\alpha$ as $n \rightarrow \infty$. One can quite easily keep track of $\Delta W_{k_\alpha}(n)$, for instance by maintaining a circular history array storing W_{k_α} over the last n steps. This approach, which we denote the *time-averaged correlation function method*, has the important advantage that one can obtain the steady state parameter sensitivity from a single simulation run.

The recipe for using the time-averaged correlation function method is then:

1. Choose a time interval Δt which is greater than the typical relaxation time of the quantity f of interest. Estimate the typical number of steps n taken in time Δt : $n = \Delta t / \bar{\tau}$.
2. For a simulation of the system in the steady state, record f and W_{k_α} every n steps (we denote each of these recordings a ‘timeslice’).
3. For each timeslice i , compute the difference between $W_{k_\alpha}^{(i)}$ and its value in the previous timeslice: $\Delta W_{k_\alpha}^{(i)} = W_{k_\alpha}^{(i)} - W_{k_\alpha}^{(i-1)}$. Compute also $f^{(i)} \Delta W_{k_\alpha}^{(i)}$.
4. Compute the averages over all timeslices of f , ΔW_{k_α} and $f \Delta W_{k_\alpha}$.
5. Calculate the correlation function $C(n) = \overline{f \Delta W_{k_\alpha}} - (\bar{f})(\overline{\Delta W_{k_\alpha}})$. As long as n is large enough this provides a measurement of $\partial \bar{f}_{\text{ss}} / \partial k_\alpha$.

Time-step pre-averaging

The time-averaged correlation function method is a convenient way to compute parameter sensitivities in a standard kinetic Monte Carlo scheme, in which both a new timestep and a new reaction channel are chosen stochastically at every step. However, choosing a new time step at every iteration is computationally expensive since it requires a random number, and is not strictly necessary for the computation of steady-state parameter sensitivity coefficients. Improved efficiency can be achieved by choosing only the new reaction channel stochastically at each iteration, and replacing δt by the mean timestep $\tau \equiv (\sum_\mu a_\mu)^{-1}$ corresponding to the current state (note that this is state dependent since it depends on the propensity functions). This amounts to *pre-averaging* over the distribution of possible time steps

for a given state of the system. It can be proved formally that if we run our simulations for a sufficiently long time, Eq. (14) is equivalent to

$$\bar{f} = \frac{\sum_{\text{steps}} f \tau}{\sum_{\text{steps}} \tau}. \quad (18)$$

Intuitively, this relation arises because a sufficiently long trajectory, under steady state conditions, will visit each state an arbitrarily large number of times and thus thoroughly sample the distribution of waiting times in each state.

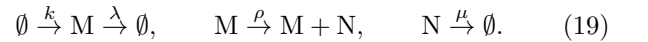
One cannot, however, compute the parameter sensitivity $\partial \bar{f}_{\text{ss}} / \partial k_\alpha$ simply by evaluating the time averages in Eq. (15) or (16) using the new definition, Eq. (18). This is because τ itself depends on the parameter k_α . Instead, a slightly more complicated expression for $\partial \bar{f}_{\text{ss}} / \partial k_\alpha$ is required; this is given in Supplementary Material Section 3. Thus, time-step pre-averaging provides a more efficient way to compute the steady-state parameter sensitivity (since the time does not need to be updated in the Monte Carlo algorithm), at the cost of a slight increase in mathematical complexity.

IV. EXAMPLES

We now apply the methods described above to three case studies: a model for constitutive gene expression for which we can compare our results to analytical theory, a simple model for a signaling pathway with stochastic focussing, and a model for a bistable genetic switch. The second and third examples are chosen because they exhibit the kind of non-trivial behaviour found in real biochemical networks, yet the state space is sufficiently compact that the parameter sensitivities can be checked using finite-state projection (FSP) methods [15]. Our implementation of the FSP methods is described more fully in Supplementary Material Section 4.

A. Constitutive gene expression

We first consider a simple stochastic model for the expression of a constitutive (unregulated) gene, represented by the following chemical reactions:



Here, M represents messenger RNA (synthesis rate k , degradation rate λ) and N represents protein (synthesis rate ρ , degradation rate μ). This model has linear propensities (as defined in Supplementary Material Section 2), which implies that the mean copy numbers $\langle M(t) \rangle$ and $\langle N(t) \rangle$ of mRNA and protein respectively obey the chemical rate equations

$$\frac{d\langle M \rangle}{dt} = k - \lambda \langle M \rangle, \quad \frac{d\langle N \rangle}{dt} = \rho \langle M \rangle - \mu \langle N \rangle \quad (20)$$

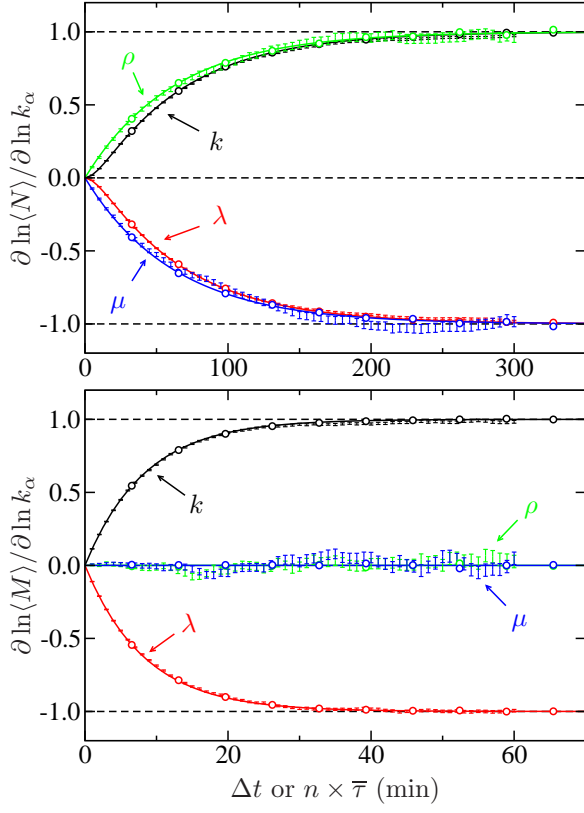


FIG. 1: (color online) Time correlation functions $C(\Delta t)$ for the sensitivity of the average protein number $\langle N \rangle_{ss}$ (top) and the average mRNA number $\langle M \rangle_{ss}$ to the model parameters, for the constitutive gene expression model in Section IV A. Points with error bars are simulations using the ensemble-average correlation function method; error bars are estimated by block-averaging (100 blocks of 10^3 trajectories; total $\approx 10^{10}$ time steps). Open circles are simulations using the time-average correlation function method with time step pre-averaging (plotted as a function of $n\bar{\tau}$); results are averages over 10 trajectories each of length 10^9 steps (in this case the error bars are smaller than symbols). Solid lines are theoretical predictions from Eq. (22). Parameters are $k = 2.76 \text{ min}^{-1}$, $\lambda = 0.12 \text{ min}^{-1}$, $\rho = 3.2 \text{ min}^{-1}$, $\mu = 0.016 \text{ min}^{-1}$, corresponding to the *cro* gene in a recent model of phage lambda [16]. For these parameters $\langle M \rangle_{ss} = 23$ and $\langle N \rangle_{ss} = 4600$.

from which follow the steady state mean copy numbers:

$$\langle M \rangle_{ss} = \frac{k}{\lambda}, \quad \langle N \rangle_{ss} = \frac{\rho k}{\lambda \mu}. \quad (21)$$

For this problem, steady-state sensitivity coefficients can be computed analytically by taking derivatives of Eqs. (21) with respect to the parameters of interest. Moreover, as shown in Supplementary Material Section 2, explicit expressions can also be found for the components of the correlation functions defined by Eqs. (12)

and (13):

$$\begin{aligned} \langle M \Delta W_{\ln k} \rangle_{ss} &= -\langle M \Delta W_{\ln \lambda} \rangle_{ss} = \langle M \rangle_{ss} (1 - e^{-\lambda \Delta t}), \\ \langle M \Delta W_{\ln \rho} \rangle_{ss} &= \langle M \Delta W_{\ln \mu} \rangle_{ss} = 0, \\ \langle N \Delta W_{\ln k} \rangle_{ss} &= -\langle N \Delta W_{\ln \lambda} \rangle_{ss} \\ &= \langle N \rangle_{ss} [\lambda(1 - e^{-\mu \Delta t}) - \mu(1 - e^{-\lambda \Delta t})] / (\lambda - \mu), \\ \langle N \Delta W_{\ln \rho} \rangle_{ss} &= -\langle N \Delta W_{\ln \mu} \rangle_{ss} = \langle N \rangle_{ss} (1 - e^{-\mu \Delta t}), \end{aligned} \quad (22)$$

where for notational convenience we consider the sensitivity with respect to the logarithm of the parameter value (e. g. $W_{\ln k} \equiv k W_k$).

Figure 1 shows the time correlation functions of Eqs. (12) and (13), computed over multiple stochastic simulation runs using the ensemble-averaged correlation function method, together with the analytical results of Eq. (22) (solid lines). The agreement between the analytic theory and simulation results is excellent. The time correlation functions converge to the expected steady state sensitivity coefficients (horizontal lines in Fig. 1). For the protein correlation functions $(\partial \ln \langle N \rangle_{ss} / \partial \ln k_\alpha)$, this occurs on a timescale governed by the relaxation rate of protein number fluctuations $1/\mu \approx 60 \text{ min}$, while the mRNA correlation functions $(\partial \ln \langle M \rangle_{ss} / \partial \ln k_\alpha)$ reach their asymptotic values on a timescale governed by the mRNA decay rate $1/\lambda \approx 8 \text{ min}$.

Figure 1 (open circles) also shows the same correlation functions, computed instead from a single stochastic simulation run, using the time-averaged correlation function method, with time-step pre-averaging. Although this method gives correlation functions (Eqs. (16) and (17)) in terms of the number of steps n in the history array, rather than the time difference Δt , these can be converted to time correlation functions by multiplying n by the expected *global* mean time step $\bar{\tau}$ (the average over states of the state-dependent mean time step τ).

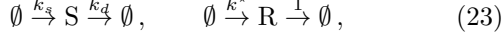
Comparing the results of the ensemble-averaged and time-averaged correlation function methods in Fig. 1 we see that the two methods give essentially the same results, but the time-averaged method produces greater accuracy (smaller error bars), for the same total number of simulation steps. Moreover, because we have used time-step pre-averaging with the time-averaged correlation function method, each simulation step is computed approximately twice as fast as in the original kinetic Monte Carlo algorithm, since one does not need to generate random numbers for the time steps [17].

B. Stochastic focusing

We now turn to a more sophisticated case study, based on the stochastic focusing model of Paulsson *et al.* [18]. In this biochemical network, a input signal molecule S downregulates the production of an output signal (or response) molecule R. Stochastic fluctuations play a crucial role, making the output much more sensitive to changes

in the input than would be predicted by a deterministic (mean-field) model.

Our reaction scheme, given in Eq. (23), contains just two chemical species, S and R. The production and degradation of S (the input signal) are straightforward Poisson processes with rates k_s and k_d respectively. The production of R (the output signal) is negatively regulated by S, and its degradation rate is set to unity to fix the time scale. Thus we have



where we use a Michaelis-Menten-like form to represent the negative regulation: $k^* = k_p/(S + K)$, with S being the copy number of the input signal molecule. Taking a mean-field approach, we might suppose that the average copy numbers $\langle S \rangle$ and $\langle R \rangle$ should obey the chemical rate equations

$$\frac{d\langle S \rangle}{dt} = k_s - k_d \langle S \rangle, \quad \frac{d\langle R \rangle}{dt} = \frac{k_r}{\langle S \rangle + K} - \langle R \rangle. \quad (24)$$

and that therefore the steady state copy numbers should be given by

$$\langle S \rangle_{ss} = \frac{k_s}{k_d}, \quad \langle R \rangle_{ss} = \frac{k_r}{\langle S \rangle_{ss} + K}. \quad (25)$$

In reality, while Eq. (25) gives the correct result for the mean input signal $\langle S \rangle_{ss}$, it is manifestly *incorrect* for the mean output signal $\langle R \rangle_{ss}$. For example for

$$k_s = 500, \quad k_d = 100, \quad k_r = 900, \quad K = 0.09 \quad (26)$$

we find from kinetic Monte Carlo simulations $\langle S \rangle_{ss} = 5$, as predicted by Eq. (25), but $\langle R \rangle_{ss} \approx 290$, whereas Eq. (25) predicts $k_r/(\langle S \rangle_{ss} + K) = 176.82$. This failure of the mean-field prediction arises because of the non-linearity of the Michaelis-Menten-like form of the production propensity for R.

Our aim is to compute the *differential gain*,

$$g = \frac{\partial \ln \langle R \rangle_{ss}}{\partial \ln \langle S \rangle_{ss}}. \quad (27)$$

which describes the local steepness of the signal-response relation ($\langle R \rangle_{ss} \sim \langle S \rangle_{ss}^g$). The gain measures the sensitivity of the system's output $\langle R \rangle_{ss}$ to its input $\langle S \rangle_{ss}$; this can be computed by measuring the sensitivities of $\langle R \rangle_{ss}$ and $\langle S \rangle_{ss}$ to the production and degradation rates of the signal molecule. Let us suppose that the signal $\langle S \rangle_{ss}$ is varied by changing its production rate k_s infinitesimally at fixed degradation rate k_d (we could have chosen instead to vary k_d or, in principle, both k_s and k_d). The gain is then

$$g = \frac{\partial \ln \langle R \rangle_{ss} / \partial k_s}{\partial \ln \langle S \rangle_{ss} / \partial k_s} = \frac{k_s}{\langle R \rangle_{ss}} \frac{\partial \langle R \rangle_{ss}}{\partial k_s} \quad (28)$$

where the second equality follows from the fact that $\partial \ln \langle S \rangle_{ss} / \partial k_s = 1/k_s$, since $\langle S \rangle_{ss} = k_s/k_d$. We use the

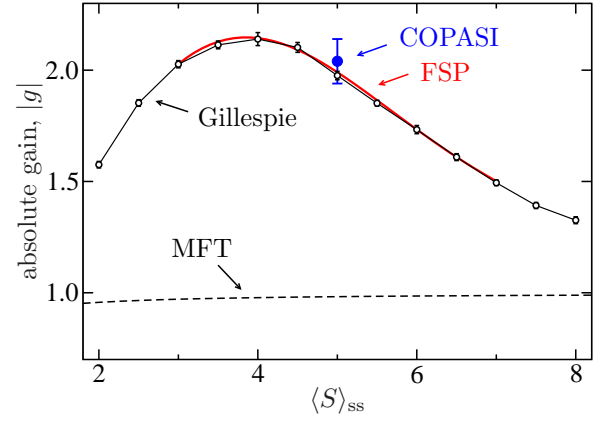
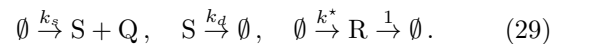


FIG. 2: (color online) Steady state absolute differential gain for the stochastic focusing model in Section IV B, where k_s is varied to control $\langle S \rangle_{ss}$, with other parameters as in Eq. (26). Open circles are Gillespie simulations using the time-average correlation function method with time step pre-averaging; error bars are estimated by averaging over 10 trajectories of length 10^8 steps. The history array length was $n = 2 \times 10^4$. The filled circle (blue) is from a Gibson-Bruck simulation in COPASI using the ensemble-average correlation function method; error bars are from block averaging (10 blocks of 10^3 samples). The thick solid line (red) is the numerical result from the finite state projection (FSP) algorithm. The dashed line is the mean-field theory (MFT) prediction.

methods described in Section III to compute the steady-state sensitivity $\partial \langle R \rangle_{ss} / \partial k_s$, and hence the gain g .

Figure 2 shows the absolute differential gain $|g|$ computed using the time-averaged correlation function method, with time-step pre-averaging, as a function of the signal strength $\langle S \rangle_{ss}$, as k_s is varied (note that in this region the actual gain is negative so $|g| = -g$). The results are in excellent agreement with the finite state projection method (FSP, see Supplementary Material Section 4). Fig. 2 (dashed line) also shows the mean-field theory prediction derived from the second of Eqs. (25), namely $g = \langle S \rangle_{ss} / (\langle S \rangle_{ss} + K)$. Stochastic focusing, as predicted by Paulsson *et al* [18], is clearly evident: the gain is much greater in magnitude for the stochastic model than the mean-field theory predicts, implying that fluctuations greatly increase the sensitivity of the output signal to the input signal [19].

In this example, the parameter of interest (k_s) is the rate constant of a single reaction (production of S). As discussed in Section II, this implies that the parameter sensitivity can be computed simply by counting the number of times this reaction is visited, which can be achieved by modifying the reaction scheme to



then computing the weight function

$$W_{k_s} = \frac{1}{k_s} [Q(t) - k_s t]. \quad (30)$$

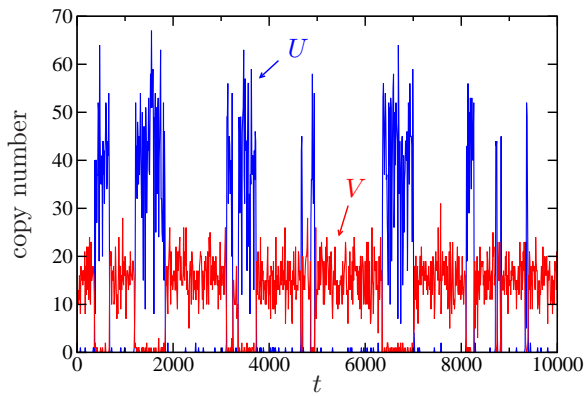


FIG. 3: (color online) Representative time traces of U and V in the Gardner *et al.* genetic switch model (Section IV C). Parameters are $\alpha_1 = 50$, $\beta = 2.5$, $\alpha_2 = 16$, and $\gamma = 1$.

(which is the analogue of Eq. (11)), and using this to obtain the relevant time-correlation functions. This requires no changes to the simulation algorithm, making it easy to use with existing software packages. As a demonstration, we computed the differential gain for the parameters in Eq. (26), using the open source simulation package COPASI [14]. To achieve this, we used the Gibson-Bruck algorithm (as implemented in COPASI) to generate samples of W_{k_s} and R at equi-spaced time points with a spacing $\Delta t = 10$ time units (chosen to be longer than the expected relaxation time of the output signal, set by the decay constant for R). By taking the difference between successive time points we compute ΔW_{k_s} and hence the correlation function defined in Eq. (16). The result, shown in blue in Fig. 2, is in good agreement with our other calculations.

C. A bistable genetic switch

As a final example, we consider a model for a bistable genetic switch, of the type constructed experimentally by Gardner *et al.* [20], in which two proteins U and V mutually repress each other's production. We suppose that transcription factor binding to the operator is cooperative, and can be described by a Hill function; the rate of production of protein U is then given by $\alpha_1/(1 + V^\beta)$ while the rate of production of V is given by $\alpha_2/(1 + U^\gamma)$ (here α_1 and α_2 describe the maximal production rates while β and γ are the Hill exponents, describing the degree of cooperativity). The units of time are fixed by setting the degradation rates of U and V to unity. For a suitable choice of parameter values, stochastic simulations of this model show switching between a U -rich state and a V -rich state, as illustrated in Fig. 3; the steady-state probability distribution P_q for the quantity $q \equiv U - V$ is bimodal, as shown in Fig. 4. We use this example to illustrate the computation of parameter sensitivities for more complicated scenarios where the system property

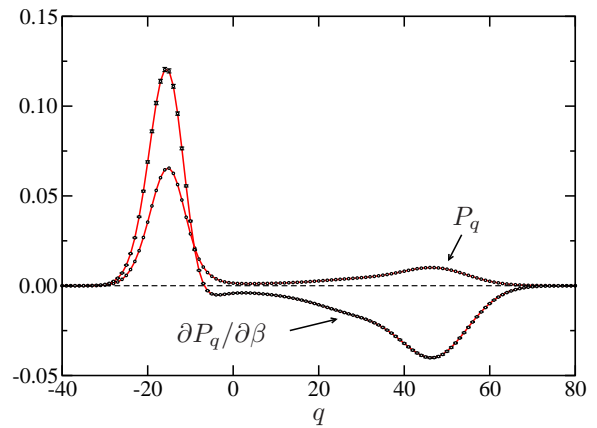
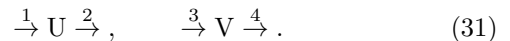


FIG. 4: (color online) Steady state probability distribution for the order parameter $q \equiv U - V$ in the Gardner *et al.* switch, and the sensitivity to β . Points (with small error bars in the case of the sensitivity) are from Gillespie simulations using the time-average correlation function method with time step pre-averaging; error bars are estimated by averaging over 10 trajectories of 10^9 steps. The history array length was $n = 5 \times 10^4$. The solid lines (red) are the results of FSP applied to this problem. Model parameters are as in Fig. 3.

of interest is not simply a mean copy number and the parameter of interest is not a simple rate constant. In particular we compute the sensitivity of the steady-state probability distribution P_q to the Hill exponent β .

Our model consists of the following reaction scheme:



in which proteins U and V are created and destroyed with propensities given by:

$$a_1 = \frac{\alpha_1}{1 + V^\beta}, \quad a_2 = U, \quad a_3 = \frac{\alpha_2}{1 + U^\gamma}, \quad a_4 = V \quad (32)$$

Let us first suppose we wish to compute $\partial P_q / \partial \beta$ using the time-averaged correlation function method, without time step pre-averaging. We use the propensity functions in Eqs. (32) to run a standard kinetic Monte Carlo (Gillespie) simulation, choosing at each step a next reaction and a time step δt . At each simulation step, we also compute the quantity

$$z \equiv \frac{\partial \ln a_1}{\partial \beta} = -\frac{V^\beta \ln V}{1 + V^\beta} \quad (33)$$

and update the weight function W according to Eq. (10), *i. e.* if reaction 1 is chosen as the next reaction, we increment W by $z(1 - a_1 \delta t)$, otherwise, we increment W by $-z a_1 \delta t$ (note that it is correctly a_1 that features in this, irrespective of the chosen next reaction). We keep track not only of the current value of W , but also of its value a fixed number n steps ago. At the same time, we keep track of the function of interest (denoted f in Sections II and III). Because we are computing the parameter sensitivity of the *distribution* P_q , we have a function f_q , and a

time-correlation function $C_{q,\text{time av}}(n)$, for *each* value of q . At each simulation step, we check the current value of $q(t)$. The function f_q is unity if $q = q(t)$ and zero otherwise (*i.e.* $f_q = \delta_{q(t),q}$). For each value of q , we then compute the time correlation function $C_{q,\text{time av}}(n)$ as prescribed in Eq. (16). As long as $n\bar{\tau}$ (where $\bar{\tau}$ is the global average time step) is longer than the typical relaxation time of the system, $C_{q,\text{time av}}(n)$ should give a good estimate for $\partial P_q/\partial\beta$.

If we are instead using time step pre-averaging, we employ a slight modification of the above procedure. At each simulation step, we choose a next reaction, but we do not choose a time step δt . In our update rules for W , we replace δt by the state-dependent mean timestep τ where $\tau^{-1} = \sum_{\mu=1}^4 a_{\mu}$. As well as keeping track of W and f_q we also need to compute at each step

$$\frac{\partial \tau}{\partial \beta} = a_1 \frac{\partial \tau}{\partial a_1} \cdot \frac{\partial \ln a_1}{\partial \beta} = -\tau^2 a_1 z. \quad (34)$$

This quantity is then used to compute $C_{q,\text{time av}}(n)$ and hence $\partial P_q/\partial\beta$ using the modified algorithm given in Supplementary Material Section 3.

An important technical point here concerns the relaxation time of the system, or the number of steps n over which we need to remember the system's history in order that the correlation function $C(n)$ gives a good estimate of the steady-state parameter sensitivity. For the previous examples studied, this timescale was given by the slowest decay rate (typically that of the protein molecules). The genetic switch, however, shows dynamical switching behaviour on a timescale that is much longer than the protein decay rate (see for example Fig. 3). We therefore need to choose a value of n such that $n\bar{\tau}$ is longer than the typical switching time. Kinetic Monte-Carlo simulations (like those in Fig. 3) show that for our model, the typical time between switching events is approximately 160 time units, while the global average time step $\bar{\tau} \approx 0.02$. The typical number of steps per switching event is therefore $160/0.02 = 8 \times 10^3$. Our chosen value of n should be at least this large. In practice we find that the correlation functions are fully converged (to within a reasonable accuracy) by $n = 5 \times 10^4$ steps (≈ 6.3 switching events), but not quite converged by $n = 2 \times 10^4$ steps (≈ 2.5 switching events). These lengthy convergence times mean that much longer simulations are needed to obtain good statistical estimates for the parameter sensitivity in this model than in the previous examples.

Figure 4 shows the steady state probability distribution P_q together with its sensitivity $\partial P_q/\partial\beta$, computed using the time-averaged correlation function method with time step pre-averaging, for the same parameters as in Fig. 3. This method gives results in excellent agreement with FSP. P_q has the bimodal shape typical of a stochastic genetic switch, with a large peak at $q \approx -15$ and a much broader peak around $q \approx 45$, with a minimum around $q \approx 5$. The sensitivity coefficient $\partial P_q/\partial\beta$ measures how the behaviour of the switch depends on the

cooperativity β of binding of the transcription factor V. We see that increasing β leads to an increased peak at $q < 0$, and a decreased peak at $q > 0$, in other words the switch spends more time in the V-rich state. Also the minimum around $q \approx 5$ decreases, suggesting that the switching frequency decreases as β increases. This is confirmed by further study using the ensemble-averaged correlation function method of the sensitivity coefficient of the switching frequency to changes in β ; the details of this will be presented elsewhere.

V. DISCUSSION

In this paper, we have shown how trajectory reweighting can be used to compute parameter sensitivity coefficients in stochastic simulations without the need for repeated simulations with perturbed values of the parameters. The methods presented here are simple to implement in standard kinetic Monte Carlo (Gillespie) simulation algorithms and in some cases can be used without any changes to the simulation code, making them compatible with packages such as COPASI [14]. For computation of time-dependent sensitivity coefficients, the method involves tracking a weight function (which depends on the derivative of the propensities with respect to the parameter of interest) and computing its covariance with the system property of interest, at the time of interest, across multiple simulations. For computing time-independent steady-state parameter sensitivities, we show that the sensitivity coefficient can be obtained as the long-time limit of a time correlation function, which can be computed either across multiple simulations (ensemble-averaged correlation function method), or as a time average in a single simulation run (time-averaged correlation function method). We further show that time step pre-averaging removes the need to choose a new time step at each simulation step, significantly improving computational efficiency. In either the time-dependent or the time-independent case, it is a trivial matter to compute multiple sensitivity coefficients (e.g. with respect to different parameters) at the same time – one simply tracks each of the corresponding weight functions simultaneously.

In deterministic models, parameter sensitivity coefficients can be computed by simultaneous integration of a set of *adjunct* ODEs, alongside the set of ODEs describing the model (see Eq. (2)). We consider the trajectory reweighting approach described here to be the exact stochastic analogue of the adjunct ODE method; the integration of the adjunct ODEs alongside the original ODEs is directly analogous to the procedure of generating a trajectory weight alongside the normal trajectory in a kinetic Monte-Carlo scheme. Indeed, one can derive an *adjunct chemical master equation* by taking the derivative of the chemical master equation with respect to the parameter of interest; it turns out that the trajectory reweighting scheme is essentially a stochastic solu-

tion method for the adjunct master equation [21].

In Section IV B, we demonstrated the use of trajectory reweighting to compute parameter sensitivities, and hence the differential gain, for a model of a stochastic signaling network. We believe that this approach has widespread potential application to signaling pathways, because it can be implemented for any existing model without any modifications to the underlying kinetic Monte-Carlo simulation code. As long as a stochastic input signal is generated by a process $\emptyset \rightarrow S \rightarrow \emptyset$, one can use the ghost particle trick to compute the sensitivity of any quantity of interest to the input signal (controlled by varying the rate k_s of the signal production reaction) by modifying the production step to $\emptyset \rightarrow S + Q$, computing the weight function from $W_{k_s} = Q/k_s - t$ (see Eq. (30)) and computing the appropriate correlation function for its covariance with the system property of interest. As proof-of-principle we calculated the differential gain for the model in Section IV B (see Fig. 2), using COPASI [14] to generate the simulation data, and a standard spreadsheet package to compute the correlation function.

In Section IV C, we used the methodology to compute the sensitivity of the probability distribution function for a bistable genetic switch, to the degree of cooperativity (Hill exponent) of binding of one of its transcription factors. This example demonstrates that trajectory reweighting is not a panacea for all problems. The bistable genetic switch has a long relaxation time, which requires the correlation function of the weight to be computed over long times, with a corresponding need for large sample sizes to obtain good statistical sampling. While trajectory reweighting works for this example, preliminary attempts to compute the parameter dependence of the switching rate show that finite differencing may be more efficient. In fact Plyasunov and Arkin [5] already discuss in which cases it may be more efficient to use finite-differencing. Because of their long relaxation

times, genetic switches are notoriously difficult to study in stochastic simulations [22]. A plethora of sophisticated schemes have been developed to address this problem [23–25], some of which could perhaps be extended to incorporate trajectory reweighting.

The present study considers how to compute parameter sensitivity coefficients—*i. e.* first derivatives of system properties with respect to the parameters. The same approach can, however, easily be used to compute higher derivatives, such as the Hessian matrix, as discussed in Supplementary Material Section 1. This raises the possibility of combining the present methods with gradient-based search algorithms, to make a sophisticated *parameter estimation* algorithm for stochastic modeling. This would offer a novel approach to a major class of problems in systems biology.

To summarise, we believe the trajectory reweighting schemes presented here are an important and useful addition to the stochastic simulation toolbox. Further research should address in detail their performance with respect to existing methods [4–6] and their application to challenging models such as those with long relaxation times, as well as their potential for use in more sophisticated parameter search algorithms.

Acknowledgments

The authors thank Mustafa Khammash for detailed advice about the FSP method and assistance with its implementation. RJA was supported by a Royal Society University Research Fellowship. The collaboration leading to this work was facilitated by the StoMP research network under BBSRC grant BB/F00379X/1 and by the e-Science Institute under theme 14: “Modelling and Microbiology”.

-
- [1] N. G. van Kampen, *Stochastic processes in physics and chemistry* (Elsevier, Amsterdam, 1992).
 - [2] D. T. Gillespie, *J. Phys. Chem.* **81**, 2340 (1977).
 - [3] M. A. Gibson and J. Bruck, *J. Phys. Chem. A* **104**, 1876 (2000).
 - [4] D. Kim, B. J. Debusschere, and H. N. Najm, *Biophys. J.* **92**, 379 (2007).
 - [5] A. Plyasunov and A. P. Arkin, *J. Comp. Phys.* **221**, 724 (2007).
 - [6] M. Rathinam, P. W. Sheppard, and M. Khammash, *J. Chem. Phys.* **132**, 034103 (2010).
 - [7] B. Harland and S. X. Sun, *J. Chem. Phys.* **127**, 104103 (2007).
 - [8] H. Kuwahara and I. Mura, *J. Chem. Phys.* **129**, 165101 (2008).
 - [9] M. K. Roh, D. T. Gillespie, and L. R. Petzold, *J. Chem. Phys.* **133**, 174106 (2010).
 - [10] D. Frenkel, B. Smit, *Understanding Molecular Simulation*, 2nd Ed. (Academic Press, San Diego, 2002).
 - [11] The fact that the two terms in Eq. (9) cancel on average could in principle be used to simplify Eq. (6) to $\partial \langle f \rangle / \partial k_\alpha = \langle f W_{k_\alpha} \rangle$. However, in practice we have found that retaining the term in $\langle W_{k_\alpha} \rangle$ improves the statistical sampling (for example if f is a constant then the right hand side of Eq. (6) vanishes exactly, as it should, even if the sampling error causes $\langle W_{k_\alpha} \rangle \neq 0$).
 - [12] If this is not the case one can always make it so by augmenting the parameter set.
 - [13] D. C. Wylie, Y. Hori, A. R. Dinner, and A. K. Chakraborty, *J. Phys. Chem. B* **110**, 12749 (2006).
 - [14] S. Hoops, *et al.*, *Bioinformatics* **22**, 3067 (2006).
 - [15] B. Munsky and M. Khammash, *J. Chem. Phys.* **124**, 044104 (2006).
 - [16] M. Santillán and M. C. Mackey, *Biophys. J.* **86**, 75 (2004).
 - [17] The efficiency gain obtained by time-step pre-averaging depends of course on the balance between computing

- time steps and choosing reactions: this might differ for different algorithms and reaction schemes.
- [18] J. Paulsson, O. G. Berg, and M. Ehrenberg, *Proc. Natl. Acad. Sci. (USA)* **97**, 7148 (2000).
- [19] For this model we find that it does not make any detectable difference whether g is computed at fixed k_d (as here) or fixed k_s . This is almost certainly not generally true, and in the present case is likely connected to the fact that the noise in S is uncorrelated with the noise in P , in the sense of S. Tanăsa-Nicola, P. B. Warren, and P. R. ten Wolde, *Phys. Rev. Lett.* **97**, 068102 (2006).
- [20] T. S. Gardner, C. R. Cantor, and J. J. Collins, *Nature* **403**, 339 (2000).
- [21] R. J. Allen and P. B. Warren, to be published elsewhere.
- [22] M. J. Morelli, P. R. ten Wolde, and R. J. Allen, *Proc. Natl. Acad. Sci. USA* **106**, 8101 (2009).
- [23] R. J. Allen, P. B. Warren, and P. R. ten Wolde, *Phys. Rev. Lett.* **94**, 018104 (2005).
- [24] R. J. Allen, C. Valeriani, and P. R. ten Wolde, *J. Phys. Condens. Matter* **21**, 463102 (2009).
- [25] A. Dickson and A. R. Dinner, *Annu. Rev. Phys. Chem.* **61**, 441 (2010).
- [26] P. B. Warren, S. Tanăsa-Nicola, and P. R. ten Wolde, *J. Chem. Phys.* **125**, 144904 (2006).
- [27] R. B. Lehoucq and D. C. Sorensen, *SIAM. J. Matrix Anal. & Appl.* **17**, 789 (1996).

SUPPLEMENTARY MATERIAL

1. DERIVATIVES OF AVERAGES WITH RESPECT TO PARAMETERS

Here, we present a convenient way to compute the derivatives of average quantities with respect to the parameters of the model, that are required to arrive at Eqs. (6) and (7) in the main text. We also show that this method generalizes easily to higher derivatives.

Noting that in the perturbed system the parameter k_α has been changed to k'_α , we use Eq. (5) in the main text to write the average of the function f in the perturbed system as

$$\langle f \rangle' = \frac{\langle f R(k'_\alpha, k_\alpha) \rangle}{\langle R(k'_\alpha, k_\alpha) \rangle} \quad (35)$$

where

$$R(k'_\alpha, k_\alpha) \equiv \frac{P(k'_\alpha)}{P(k_\alpha)} = e^{\ln P(k'_\alpha) - \ln P(k_\alpha)}. \quad (36)$$

The function $R(k'_\alpha, k_\alpha)$ has the property that $\partial R / \partial k'_\alpha = W'_{k'_\alpha} R$ where $W'_{k'_\alpha} = \partial \ln P(k'_\alpha) / \partial k'_\alpha$. We then have

$$\frac{\partial \langle f \rangle'}{\partial k'_\alpha} = \frac{\langle f W'_{k'_\alpha} R \rangle}{\langle R \rangle} - \frac{\langle f R \rangle \langle W'_{k'_\alpha} R \rangle}{\langle R \rangle^2}. \quad (37)$$

Taking the limit $k'_\alpha \rightarrow k_\alpha$ (for an infinitesimal perturbation), and noting thereby that $R \rightarrow 1$ and $W'_{k'_\alpha} \rightarrow W_{k_\alpha}$, yields Eqs. (6) and (7) in the main text.

Taking this procedure further allows the computation of higher derivatives; one can show for instance that the Hessian is

$$\begin{aligned} \frac{\partial^2 \langle f \rangle}{\partial k_\alpha \partial k_\beta} &= \langle f W_{k_\alpha} W_{k_\beta} \rangle + \langle f W_{k_\alpha k_\beta} \rangle - \langle f W_{k_\alpha} \rangle \langle W_{k_\beta} \rangle - \langle f W_{k_\beta} \rangle \langle W_{k_\alpha} \rangle \\ &\quad - \langle f \rangle \langle W_{k_\alpha} W_{k_\beta} \rangle - \langle f \rangle \langle W_{k_\alpha k_\beta} \rangle + 2 \langle f \rangle \langle W_{k_\alpha} \rangle \langle W_{k_\beta} \rangle. \end{aligned} \quad (38)$$

where

$$W_{k_\alpha k_\beta} = \frac{\partial^2 \ln P}{\partial k_\alpha \partial k_\beta}. \quad (39)$$

Eq. (38) is potentially useful for gradient search algorithms. This expression is likely to simplify in many cases – for instance we expect that $\partial^2 \ln P / \partial k_\alpha \partial k_\beta$ often vanishes for $k_\alpha \neq k_\beta$. One might also use the fact that $\langle W_{k_\alpha} \rangle = \langle W_{k_\beta} \rangle = 0$, but it may improve the statistical sampling to retain these terms (see discussion in main text).

2. EXACT RESULTS FOR PROBLEMS WITH LINEAR PROPENSITIES

In this Section we describe some exact results that can be obtained for models with linear propensity functions, in particular for the correlation functions defined in Eqs. (12) and (13) in the main text. The analysis draws heavily on established literature results (which we summarize below). More details and links to earlier literature can be found in the appendix to Supplementary Ref. 26.

To fix notation, let us suppose that the α -th propensity function depends linearly on the copy numbers N_j , namely $a_\alpha = \sum_j K_{\alpha j} N_j + b_\alpha$ where $K_{\alpha j}$ and b_α are constants which we assume to be proportional to the rate constant k_α . Our aim is to compute the sensitivity coefficients $\partial \langle N_i \rangle / \partial \ln k_\alpha$.

It is well known that for linear propensity functions the moment equations close successively. Thus, the mean copy numbers $\langle N_i \rangle$ obey

$$\frac{\partial \langle N_i \rangle}{\partial t} = \sum_\alpha \nu_{i\alpha} \langle a_\alpha \rangle = \sum_j K_{ij} \langle N_j \rangle + b_i \quad (40)$$

where $\nu_{i\alpha}$ is the stoichiometry matrix (describing the change in the copy number of the i -th species due to the firing of the α -th reaction), $K_{ij} = \sum_\alpha \nu_{i\alpha} K_{\alpha j}$ and $b_i = \sum_\alpha \nu_{i\alpha} b_\alpha$. Note that K_{ij} is usually asymmetric. For the second moments, the variance-covariance matrix $S_{ij}(t) = \langle \Delta N_i(t) \Delta N_j(t) \rangle$, where $\Delta N_i = N_i - \langle N_i \rangle$, obeys

$$\frac{\partial S_{ij}}{\partial t} = \sum_k (K_{ik} S_{jk} + K_{jk} S_{ik}) + H_{ij} \quad (41)$$

where

$$H_{ij} = \sum_\alpha \nu_{i\alpha} \nu_{j\alpha} \langle a_\alpha \rangle. \quad (42)$$

Note that $S_{ij}(t)$ is symmetric. Finally the time-ordered two-point correlation functions $\langle \Delta N_i(t) \Delta N_j(t') \rangle$ with $t > t'$ obey a regression theorem

$$\frac{\partial \langle \Delta N_i(t) \Delta N_j(t') \rangle}{\partial t} = \sum_k K_{ik} \langle \Delta N_k(t) \Delta N_j(t') \rangle. \quad (43)$$

This concludes our survey of the established literature results.

We now employ the ghost particle trick of Section II in the main text, and suppose that reaction α now creates a noninteracting species Q_α , in addition to its usual products. Following Eq. (40), the mean copy number $\langle Q_\alpha \rangle$ obeys

$$\frac{\partial \langle Q_\alpha \rangle}{\partial t} = \langle a_\alpha \rangle. \quad (44)$$

For the second moments, Eq. (41) becomes

$$\frac{\partial S_{i\alpha}}{\partial t} = \sum_j (K_{ij} S_{j\alpha} + K_{\alpha j} S_{ij}) + H_{i\alpha}. \quad (45)$$

where $S_{i\alpha} = \langle \Delta N_i \Delta Q_\alpha \rangle$. The second term in this can be rewritten as

$$\sum_j K_{\alpha j} S_{ij} = \langle \Delta N_i (\sum_j K_{\alpha j} \Delta N_j) \rangle = \langle \Delta N_i \Delta a_\alpha \rangle. \quad (46)$$

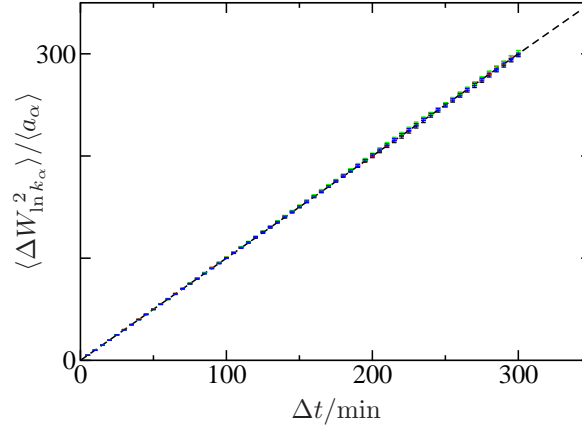


FIG. 5: (color online) Growth of variance of weight functions in steady state, normalised by the expected growth rate. Model and parameters as for Fig. 1 in the main text.

The stoichiometry matrix entry for Q_α consists of a ‘+1’ for the α -th reaction, and zero elsewhere, so that Eq. (42) becomes $H_{i\alpha} = \nu_{i\alpha}\langle a_\alpha \rangle$. Finally we have

$$\frac{\partial S_{i\alpha}}{\partial t} = \sum_j K_{ij} S_{j\alpha} + \langle \Delta N_i \Delta a_\alpha \rangle + \nu_{i\alpha} \langle a_\alpha \rangle. \quad (47)$$

By similar argumentation we also obtain

$$\frac{\partial S_{\alpha\alpha}}{\partial t} = 2\langle \Delta Q_\alpha \Delta a_\alpha \rangle + \langle a_\alpha \rangle \quad (48)$$

where $S_{\alpha\alpha} = \langle \Delta Q_\alpha^2 \rangle$.

Armed with these results, let us now turn to the problem of computing the weight function W_{k_α} . We write the continuous-time analogue of Eq. (11) in the main text:

$$k_\alpha W_{k_\alpha}(t) = Q_\alpha(t) - \int_0^t dt' a_\alpha(t') \quad (49)$$

(note that by substituting Eq. (44) into Eq. (49) we recover our previous observation that $\langle W_{k_\alpha} \rangle = 0$). Again writing $W_{\ln k_\alpha} \equiv k_\alpha W_{k_\alpha}$, it follows from Eq. (49) that

$$\langle N_i W_{\ln k_\alpha} \rangle = \langle \Delta N_i \Delta W_{\ln k_\alpha} \rangle = S_{i\alpha}(t) - \int_0^t dt' \langle \Delta N_i(t) \Delta a_\alpha(t') \rangle. \quad (50)$$

Thus, now noting explicitly the time-dependence of the various terms, we obtain

$$\frac{\partial \langle N_i W_{\ln k_\alpha} \rangle}{\partial t} = \frac{\partial S_{i\alpha}}{\partial t} - \langle \Delta N_i(t) \Delta a_\alpha(t) \rangle - \int_0^t dt' \frac{\partial \langle \Delta N_i(t) \Delta a_\alpha(t') \rangle}{\partial t}. \quad (51)$$

Exploiting the linearity of the propensity functions, the regression theorem implies

$$\frac{\partial \langle \Delta N_i(t) \Delta a_\alpha(t') \rangle}{\partial t} = \sum_j K_{ij} \langle \Delta N_j(t) \Delta a_\alpha(t') \rangle. \quad (52)$$

Hence

$$\frac{\partial \langle N_i W_{\ln k_\alpha} \rangle}{\partial t} = \frac{\partial S_{i\alpha}}{\partial t} - \langle \Delta N_i(t) \Delta a_\alpha(t) \rangle - \sum_j K_{ij} \int_0^t dt' \langle \Delta N_j(t) \Delta a_\alpha(t') \rangle. \quad (53)$$

Eliminating the time integrals between this and Eq. (50), we get

$$\frac{\partial \langle N_i W_{\ln k_\alpha} \rangle}{\partial t} = \frac{\partial S_{i\alpha}}{\partial t} - \langle \Delta N_i(t) \Delta a_\alpha(t) \rangle - \sum_j K_{ij} S_{j\alpha}(t) + \sum_j K_{ij} \langle N_j W_{\ln k_\alpha} \rangle. \quad (54)$$

Eliminating $\partial S_{i\alpha}/\partial t$ between this and Eq. (47) gives finally

$$\frac{\partial \langle N_i W_{\ln k_\alpha} \rangle}{\partial t} = \sum_j K_{ij} \langle N_j W_{\ln k_\alpha} \rangle + \nu_{i\alpha} \langle a_\alpha \rangle. \quad (55)$$

This is an ODE which give the evolution of $\langle N_i W_{\ln k_\alpha} \rangle$ in terms of known quantities. It can be compared with the adjunct ODE that is obtained by differentiating Eq. (40) with respect to $\ln k_\alpha$. The two ODEs are identical and share the same initial conditions. For this specific case, this is a direct proof of the general result in the main text, namely that $\langle N_i W_{\ln k_\alpha} \rangle \equiv \partial \langle N_i \rangle / \partial \ln k_\alpha$. Whilst this is interesting, it is not quite what we are after, which is a theory for the correlation functions defined in the main text. To find this we first note a generalisation of the regression theorem is

$$\frac{\partial \langle N_i(t) W_{\ln k_\alpha}(t_0) \rangle}{\partial t} = \sum_k K_{ik} \langle N_k(t) W_{\ln k_\alpha}(t_0) \rangle. \quad (56)$$

Subtracting this from Eq. (55) generates a set of coupled ODEs for the correlation functions $C_i(t, t_0) = \langle N_i(t) \Delta W_{\ln k_\alpha}(t, t_0) \rangle$, which should be solved with the initial conditions $C_i(t, t_0) = 0$ at $t = t_0$. In steady state these ODEs are

$$\frac{\partial C_i(\Delta t)}{\partial(\Delta t)} = \sum_j K_{ij} C_j(\Delta t) + \nu_{i\alpha} \langle a_\alpha \rangle. \quad (57)$$

The initial conditions are $C_i(0) = 0$. This is the key result of this Section, as in principle it allows for explicit calculation of the correlation functions. Comparing to the adjunct ODE obtained by differentiating Eq. (40) with respect to $\ln k_\alpha$, we see that for this case we also have a direct proof of the general result claimed in the main text, that $C_i(\Delta t) \rightarrow \partial \langle N_i \rangle / \partial \ln k_\alpha$ as $\Delta t \rightarrow \infty$. For the constitutive gene expression model in Section IV A in the main text, we solved Eq. (57) to obtain the results given in Eq. (22) in the main text.

To complete the general discussion here, let us derive an expression for the variance of $\Delta W_{\ln k_\alpha}$. From the definition in Eq. (49) we have

$$\langle W_{\ln k_\alpha}^2 \rangle = S_{\alpha\alpha} - 2 \int_0^t dt' \langle \Delta Q_\alpha(t) \Delta a_\alpha(t') \rangle + \int_0^t dt' \int_0^t dt'' \langle \Delta a_\alpha(t') \Delta a_\alpha(t'') \rangle \quad (58)$$

Thus

$$\frac{\partial \langle W_{\ln k_\alpha}^2 \rangle}{\partial t} = \frac{\partial S_{\alpha\alpha}}{\partial t} - 2 \langle \Delta Q_\alpha \Delta a_\alpha \rangle - 2 \int_0^t dt' \frac{\partial \langle \Delta Q_\alpha(t) \Delta a_\alpha(t') \rangle}{\partial t} + 2 \int_0^t dt' \langle \Delta a_\alpha(t) \Delta a_\alpha(t') \rangle. \quad (59)$$

The last two terms in this cancel, on account of the regression theorem. Further cancellations occur when Eq. (48) for $\partial S_{\alpha\alpha}/\partial t$ is inserted, giving finally $\partial \langle W_{\ln k_\alpha}^2 \rangle / \partial t = \langle a_\alpha \rangle$. Since $W_{\ln k_\alpha}(t) = W_{\ln k_\alpha}(t_0) + \Delta W_{\ln k_\alpha}$, and $W_{\ln k_\alpha}(t_0)$ and $\Delta W_{\ln k_\alpha}$ are uncorrelated, it follows that $\langle W_{\ln k_\alpha}^2(t) \rangle = \langle W_{\ln k_\alpha}^2(t_0) \rangle + \langle \Delta W_{\ln k_\alpha}^2 \rangle$. Integrating $\partial \langle W_{\ln k_\alpha}^2 \rangle / \partial t = \langle a_\alpha \rangle$ and inserting in this last expression gives $\langle \Delta W_{\ln k_\alpha}^2 \rangle = \int_{t_0}^t dt' \langle a_\alpha \rangle$. As a particular case, in steady state, $\langle \Delta W_{\ln k_\alpha}^2 \rangle = \langle a_\alpha \rangle \Delta t$. Thus we do indeed see that in steady state $\Delta W_{\ln k_\alpha}$ has a zero mean and a variance that grows linearly in time, justifying our claim that it behaves essentially like a random walk. Some results confirming this analysis are shown in Fig. 5.

3. TIME STEP PRE-AVERAGING

In the time step pre-averaging approach, we do not select time steps as part of our kinetic Monte Carlo algorithm, but instead use the state-dependent average time step $\tau \equiv (\sum_{\mu} a_{\mu})^{-1}$ in our expression for the time average of quantity f , as in Eq. (18) in the main text. For the purposes of this Section, we define a new notation:

$$E(f\tau) = \frac{\sum_{\text{steps}} f\tau}{N_{\text{steps}}} \quad (60)$$

in which the sum is over the values of the system function f multiplied by the (state-dependent) mean time step, computed at each step along a kinetic Monte-Carlo trajectory of length N_{steps} . Note that $E(f\tau)$ can be computed using an algorithm that does not keep track of time but only of the choice of reaction channel. We can then rewrite Eq. (18) in the main text as

$$\bar{f} = \frac{\sum_{\text{steps}} f\tau}{\sum_{\text{steps}} \tau} = \frac{E(f\tau)}{E(\tau)}. \quad (61)$$

When using time step pre-averaging, the correlation function Eq. (16) in the main text must be modified because the relative probability of generating a given sequence of states (Eq. (4) in the main text) takes a different form when the algorithm does not keep track of time, and because the average time step τ in Eq. (61) itself usually depends on the parameter in question.

In a kinetic Monte Carlo scheme in which the next reaction is selected as normal, but time is not tracked, the probability of generating a given trajectory is proportional to

$$P = \prod_{\text{steps}} (a_{\mu} / \sum_{\mu} a_{\mu}). \quad (62)$$

(i.e. the part of Eq. (4) in the main text concerning the time step distribution is discarded). In analogy to Eq. (5) in the main text, one can write the average $E'(f\tau)$ for the perturbed problem in terms of averages over unperturbed trajectories:

$$E'(f\tau) = \frac{E(f\tau' P' / P)}{E(P' / P)}. \quad (63)$$

Taking derivatives of Eq. (63) with respect to the parameter k_{α} as described in Section 1 above, it follows that

$$\frac{\partial E(f\tau)}{\partial k_{\alpha}} = E\left(f \frac{\partial \tau}{\partial k_{\alpha}}\right) + E(f\tau W_{k_{\alpha}}) - E(f\tau) E(W_{k_{\alpha}}) \quad (64)$$

where the fact that τ depends on k_{α} leads to an extra term (the first term) in comparison to Eq. (6) in the main text. Computing $W_{k_{\alpha}} = \partial \ln P / \partial k_{\alpha}$ using Eq. (62), it turns out that $W_{k_{\alpha}} = \sum_{\text{steps}} \delta W_{k_{\alpha}}$ has the same form as before, but with δt replaced by τ ,

$$\delta W_{k_{\alpha}} = \frac{\partial \ln a_{\mu}}{\partial k_{\alpha}} - \frac{\partial (\sum a_{\mu})}{\partial k_{\alpha}} \tau. \quad (65)$$

Since the weight function in Eq. (65) behaves like a random walk, steady-state parameter sensitivities should be computed using the correlation function trick (as in Section III in the main text). From Eq. (64) we have:

$$\frac{\partial E(f\tau)}{\partial k_{\alpha}} = E\left(f \frac{\partial \tau}{\partial k_{\alpha}}\right) + \lim_{n \rightarrow \infty} C(n) \quad (66)$$

where

$$C(n) = E(f\tau\Delta W_{k_\alpha}(n)) - E(f\tau)E(\Delta W_{k_\alpha}(n)) \quad (67)$$

with $\Delta W_{k_\alpha}(n)$ given by Eq. (17) in the main text, using the present Eq. (65) to generate W_{k_α} . Finally, the parameter sensitivity coefficient itself is given by differentiating Eq. (61),

$$\frac{\partial \bar{f}}{\partial k_\alpha} = \frac{1}{E(\tau)} \frac{\partial E(f\tau)}{\partial k_\alpha} - \frac{E(f\tau)}{E(\tau)^2} \frac{\partial E(\tau)}{\partial k_\alpha}. \quad (68)$$

The quantity $\partial E(\tau)/\partial k_\alpha$ in the second term is given by setting $f = 1$ in Eqs. (66) and (67). When using time step pre-averaging in combination with the time-averaged correlation function method, one computes the parameter sensitivity coefficient using Eq. (68) rather than simply taking the limit of the correlation function as $n \rightarrow \infty$.

We note that while Eq. (68) looks formidable, its actual computation is fairly straightforward. To obtain both \bar{f} and $\partial \bar{f}/\partial k_\alpha$, one computes trajectory averages of the set of quantities defined by $\{1, f\} \otimes \{1, \tau\} \otimes \{1, \Delta W_{k_\alpha}(n)\}$, together with $\partial \tau/\partial k_\alpha$ and $f \partial \tau/\partial k_\alpha$. These averages are calculated by summing the respective quantities along the trajectory and dividing by the number of steps.

4. THE FINITE STATE PROJECTION ALGORITHM

The master equation describes the evolution of the probability $P(N_i; t)$ that a system is in the state N_i at time t . For the sake of compactness we will adopt the notation s and s' for the states N_i and N'_i respectively. The master equation is [1]

$$\frac{\partial P_s}{\partial t} = \sum_{s'} [w_{s's} P_{s'} - w_{ss'} P_s] \quad (69)$$

where $w_{ss'}$ is the transition rate from s to s' , given by

$$w_{ss'} = \begin{cases} a_\mu(N_i) & \mu\text{-th reaction is } N_i \rightarrow N'_i, \\ 0 & \text{otherwise.} \end{cases} \quad (70)$$

The finite state projection (FSP) algorithm is a numerical solution scheme for the master equation based on the idea of truncating the state space. For full details of the original FSP algorithm we refer to the work of Munsky and Khammash [15]. Here we outline the basic principles of the scheme and the small changes needed to adapt it to the computation of steady-state sensitivity coefficients. The starting point is to note that Eq. (69) is a *linear* ODE for P , and may be written in the matrix form

$$\frac{\partial \mathbf{P}}{\partial t} = \mathbf{A} \cdot \mathbf{P} \quad (71)$$

where \mathbf{A} is an infinite-dimensional sparse matrix. To make this into a tractable numerical proposition, the FSP algorithm truncates the state space to a finite size D . The truncation is chosen so as to contain almost all of the probability $P(N_i)$ under the conditions of interest. For the problems encountered here, a (hyper-)rectangular truncation scheme works, $N_i^0 \leq N_i \leq N_i^1$, for which $D = \prod_i \Delta N_i$ where $\Delta N_i = N_i^1 - N_i^0 + 1$. The question then is how to handle the states *not* included in the truncation scheme. In the original FSP algorithm the extra states are lumped together into a single meta-state. All the transitions leaving the truncated state space are connected to this new meta-state, and all the transitions entering the truncated state space are discarded. With this approximation \mathbf{A} becomes a $(D + 1) \times (D + 1)$ sparse matrix, and one can use standard numerical

methods to exponentiate the matrix and advance the probability distribution, *i. e.* $\mathbf{P}(t) = e^{\mathbf{A}t} \cdot \mathbf{P}(0)$. The advantage of introducing the meta-state is that Munsky and Khammash can prove some sophisticated truncation theorems which provide a certificate of accuracy for the scheme.

For the present problem we are interested in the steady state probability distribution \mathbf{P}^{eq} . However the meta-state is an absorbing state, which frustrates the direct computation of \mathbf{P}^{eq} . To avoid this, we discard *all* transitions which leave or enter the truncated state space whilst, obviously, retaining all the transitions contained entirely within the truncated state space. The meta-state is then no longer needed and \mathbf{A} becomes a $D \times D$ sparse matrix. The steady state distribution is found by solving $\mathbf{A} \cdot \mathbf{P}^{\text{eq}} = 0$, in other words \mathbf{P}^{eq} is the right-eigenvector of \mathbf{A} belonging to eigenvalue zero. That such an eigenvector exists is a textbook argument [1]: conservation of probability, $\sum_s P_s = 1$, implies $\sum_s \partial P_s / \partial t = 0$ and hence $\mathbf{1} \cdot \mathbf{A} = 0$ where $\mathbf{1}$ is a row-vector with entries all equal to unity. Since therefore $\mathbf{1}$ is a *left*-eigenvector of \mathbf{A} with eigenvalue zero, it follows under mild and non-restrictive conditions [1] that there is a corresponding *right*-eigenvector of \mathbf{A} with the same eigenvalue. This is the desired steady state probability distribution.

Well-established numerical methods exist to obtain the eigenvectors of a sparse matrix. For the present problems we have used the functionality provided in MathWorks MATLAB. For an open-source solution, we have also had good success with the OCTAVE interface to ARPACK which implements an implicitly restarted Arnoldi method [27]. From a practical point of view, we find we are limited to truncated state spaces of size $D \lesssim 25\,000$ for MATLAB, and somewhat smaller for the OCTAVE interface to ARPACK. This effectively limits consideration to problems involving at most two state variables N_i ($i = 1, 2$) and motivates the choice of examples in the main text.

Once \mathbf{P}^{eq} is found we can calculate $\langle f \rangle = \sum_s f_s P_s^{\text{eq}}$. Sensitivity coefficients like $\partial \langle f \rangle / \partial k_\alpha$ are then found by solving the FSP problem at k_α and $k_\alpha + h$ and using Eq. (3) in the main text, with h typically being a few percent of k_α . Note that although the master equation describes a stochastic process, it is itself a *deterministic* ODE. Hence this method of computing sensitivity coefficients by finite differencing is appropriate. In the absence of truncation theorems, convergence is verified empirically.
